17.383 Microprocessors A
Lab 4
Fall 2010

Analog to Digital Conversion Using the PIC16F684 Microcontroller

Objectives

1) To use MPLAB IDE software, PICC Compiler, and external hardware to demonstrate the following:

    a. Analog to Digital Conversion
    b. Construct a circuit to demonstrate Analog to Digital Conversion using the PIC16F684
    c. Develop Code to utilize the PIC16F684 Analog to Digital capabilities

Materials

☐	1	Computer (PC) with an available USB Port
☐		MPLAB IDE and HI-Tech PICC Lite software installed on PC
☐	1	PICkit ™ 1 with USB cable
☐	1	PIC16F684 – supplied with PICkit ™ 1 Starter Kit
☐	1	Voltmeter
☐	1	1000 picofarad capacitor (102) ... note: AKA 1 nanofarad or 0.001 microfarad
☐	1	0.01 microfarad capacitor (103) – if needed for power circuit
☐	11	470 ohm resistors (yellow violet brown)
☐	1	1 k ohm resister (brown black red)
☐	1	10k ohm potentiometer
☐	10	LEDs *or* 10 LED Bargraph Display
☐	1	10 LED Bargraph Display *or* 10 LEDs

Voltage Regulator Circuit (*Parts only needed if not previously constructed*)

☐	1	Breadboard
☐	1	Oscilloscope
☐	1	Voltmeter
☐	1	DC Power Supply or 9 volt battery
☐	1	9 volt battery clip (if using a 9 volt battery)
☐	1	LM7805 Voltage Regulator
☐	1	0.33 microfarad capacitor (334)
☐	1	0.1 microfarad capacitor (104)
☐	1	470 ohm resistors (yellow violet brown)
☐	1	LED

WARNINGS AND PRECAUTIONS

1) Never remove the PIC16F684 from an energized circuit

2) Do not construct circuits while energized

3) Follow electrical safety precautions

Source File Locations

1) HI-TECH Universal Toolsuite

   - HI-TECH ANSI C Compiler

     C:\Program Files\HI-TECH Software\9.70\bin\picc.exe

2) Microchip MPASM Toolsuite

   - MPASM Assembler (mpasmwin.exe)

     C:\Program Files\Microchip\MPASM Suite\mpasmwin.exe

   - MPLINK Object Linker (mplink.exe)

     C:\Program Files\Microchip\MPASM Suite\mplink.exe

   - MPLIB Librarian (mplib.exe)

     C:\Program Files\Microchip\MPASM Suite\mplib.exe

Background Information

Before we get into the specifics of the PIC16F684 Analog to Digital Converter (ADC) converter lets focus on the fundamentals of A/D conversions.

First … an analog signal is continuous in amplitude & time within certain limits, i.e., it changes smoothly without interruptions. An example is a sinusoidal signal.

A Digital signal is discrete in amplitude and time … i.e., it can only take certain specific values within certain limits at specific time intervals. When numbers are assigned to these steps (usually binary numbers) the result is a digital signal. An example is a

square wave, a 1-Bit digital signal with its high level being a binary '1' and its low level being a binary '0'.

An Analog-to-Digital converter (ADC) is an electronic circuit that changes or converts a continuous analog signal into a digital signal without altering its critical content. An Analog-to-Digital converter samples an analog waveform at uniform time intervals and assigns a digital value to each sample.

Sampling is the process of analyzing the continuous analog signal with measurements taken at discrete and standard intervals. In conjunction with sampling … the device needs to be able to "hold" the signal for a finite amount of time. During the "hold" time, the ADC will perform its function of converting the signal from analog to digital. The "hold" is performed via a storage capacitor. Up to the time the "hold" is commanded, the capacitor is tracking/sampling the analog signal.

The PIC16F684 Analog-to-Digital converter (A/D) allows conversion of an analog input signal to a 10-bit binary representation of that signal. The PIC16F684 has eight analog inputs, multiplexed into one sample and hold circuit. The output of the sample and hold is connected to the input of the converter. The converter generates a binary result via successive approximation and stores the result in a 10-bit register. The voltage reference used in the conversion is software selectable to either VDD or a voltage applied by the VREF pin.

There are three registers available to control the functionality of the A/D module:

ANSEL (Register 9-1)
ADCON0 (Register 9-2)
ADCON1 (Register 9-3)

The A/D conversion can be supplied in two formats, either left or right shifted. The ADFM bit (ADCON0<7>) controls the output format.

Steps to follow for analog to digital conversion:

1. Configure the A/D module
2. Configure A/D interrupts (if desired)
3. Wait the required acquisition time
4. Start conversion
5. Wait for A/D conversion to complete
6. Read A/D Result register pair
7. For the next conversion … go to step 1 or step 2 as required

17.383 Microprocessors A
Lab 4
Fall 2010

This lab will use the PIC16F684 ADC function. The results will be verified by comparing the actual input to the LED converted value.

Pre-Lab Preparation

- Develop the required code as outlined in the procedural steps
- Construct the required circuit on your breadboard

Procedure

Experiment 1. VOLTAGE REGULATOR CIRCUIT

a. If not already constructed from a previous lab, construct the voltage regulator circuit as outlined in Experiment 1.b. through 1.h, otherwise go to Experiment 2.

b. Review lab Warnings and Precautions.

c. Construct the schematic shown in Figure 1 on your breadboard. **DO NOT INCLUDE JP2.** Some guidelines for your construction follows:

1. Locate your voltage regulator circuit in a corner of your board as this circuit will be required throughout the semester.

d. Apply 9 volts to the VIN connection.

e. Using the voltmeter and an oscilloscope, verify that the +5V_REG port reads +5 volts and the output is clean and free of interference.

f. Power down the circuit.

g. Construct the circuit shown in Figure 2. This will be a visual aid in determining if voltage is present (when the LED is turned on).

h. Apply 9 volts to VIN connection and verify that the LED lights.

i. Power down the circuit.

Experiment 2. ANALOG TO DIGITAL CONVERSION

a. Develop a MPLAB IDE Project using the Code provided in Figure 5 which is also contained in the text file called Lab_4A.txt located on the course website.

The Code provided will execute as follows:

1. Performs Analog to Digital Conversion using the PIC16F684 and the circuitry of the PICkit 1 Starter Kit.

2. PICkit 1 Starter Kit LEDs will display the digital result of the ADC operation. The Most Significant Bit (MSB) is represented by LED D7. This corresponds to 2^{10}. The two Least Significant Bits (LSB) will not be displayed; therefore, D0 represents the 2^2 bit. We are ignoring the two least significant bits in this portion of the lab. PICKit 1 Starter Kit LED configuration is shown in Figure 3.

3. The PICkit 1 Starter Kit potentiometer RP1 is connected to RA0 and will be used to vary the voltage to the ADC (Figure 4).

b. Examine the code provided along with the schematic shown in Figure 4. Understand how the Code was developed, what features of the PIC16F684 are turned on/off as well as what the expected voltages will be seen by the ADC.

c. Modify the Code provided so that your code contains the following:

1. Code written in “C”. Use the PICC software to produce the code files.

2. Code contains header information as shown in Figure 6.

3. Save the program as Lab_4A.c.

4. Build you code. Once successful, burn the code to the PIC16F684 using the PICkit 1 Starter kit.

d. Turn RP1 counter clock wise (CCW) until it hits the stops. All LEDs should be extinguished at this point.

e. Slowly turn RP1 clock wise (CW). You should see the LEDs start to turn on/off. The binary representation of the LEDs will increase until all the

LEDs are on when RP1 hits the other stop.

f. Turn RP1 CCW until it hits the stops again. All LEDs should be extinguished at this point.

g. Measure the voltage from RA0 (pin 13) to ground or at RP1 pin 2.

h. Record the voltage from Experiment 2.g. in Table 1. Table 1 is also provided in Excel format on the Class Web Site.

i. Rotate RP1 CW until your voltmeter reads 0.0 volts (NOTE: FULL CCW and 0.0 may be the same position). Record the state of each LED, the actual voltage on your voltmeter, and calculate the voltage by using the LED display.

j. Repeat Experiment 2.i. for all the values listed in Table 1.

k. Insert a copy of your code, Lab_4A.c at the end of the lab report in the space provided as well as email a copy.

TABLE 1

Voltage (Volts)	LED State ("1" or "0")								Actual Voltage	Calculated Voltage
	D7	**D6**	**D5**	**D4**	**D3**	**D2**	**D1**	**D0**		
FULL CCW										
0.0										
0.5										
1.0										
1.5										
2.0										
2.5										
3.0										
3.5										
4.0										
4.5										
5.0										
FULL CW										

l. **<u>QUESTION:</u>** How accurate was the ADC as compared to the actual voltages applied to Pin 13 or RP1 pin 2?

Experiment 3. ANALOG TO DIGITAL CONVERSION USING ALL 10 BITS

a. Construct your own circuit that will utilize all 10 bits from the ADC and an analog input (0 to 5 volt DC) of the same type circuit shown in Figure 4. Circuit can be constructed such that it utilizes the LED configuration shown in Figure 3, however, you can construct the circuit as you desire. A circuit diagram shall be submitted with your Lab Report (hand drawn is acceptable). The following precautions need to be observed:

1. Protect the LEDs and PIC16F684 by using 450 ohm resisters, vice the 150 ohm resisters shown in Figure 3.

b. Modify the Code from Experiment 2 or write your own Code that will perform the operations described in Experiment 3.a. Save the program as Lab_4B.c.

c. Construct a Table similar to that of Table 1 to verify proper operation of your circuit and your code.

d. When the hardware and software are functioning as desired, do the following:

1. Notify the instructor so the operation can be observed. Have the instructor initial your Lab Notebook in the space you provided.

2. Insert a copy of your code, Lab_4B.c at the end of the lab report in the space provided as well as email a copy.

Summary:

The microcontroller can be used for more than just digital input/output as demonstrated by this lab. The PIC16F684 has eight analog to digital converters available for use. The conversion performed by the device provides an accurate conversion of the analog value. Ease of use is also a plus.

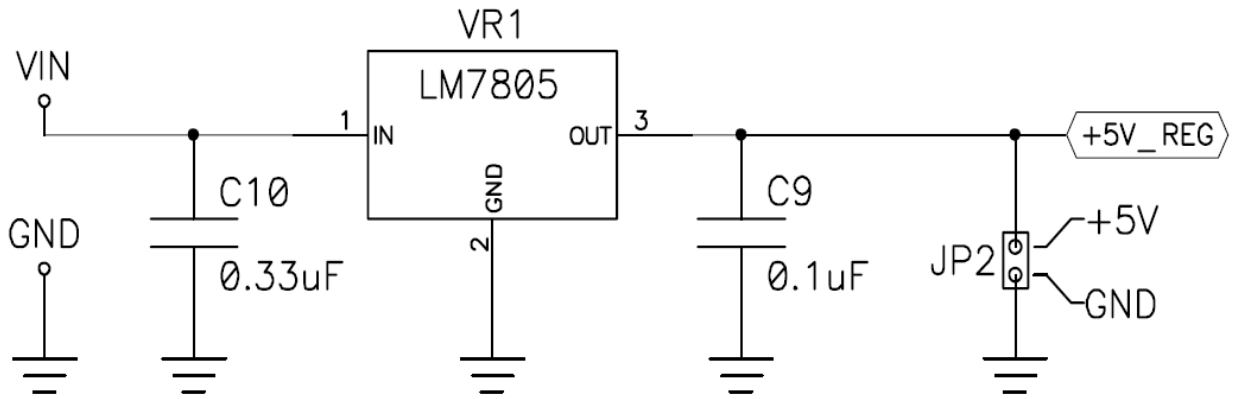
In real world applications we see that sensors provide analog outputs. Without the ability to convert these signals to a digital format would make processing these signals more complex.

We have utilized both analog and digital applications in the process of performing this lab.

Lab Report:

1. Use template provided on the Class Web Site.

Figure 1 – Voltage Regulator Circuit



7805 Pin Connections - Top View

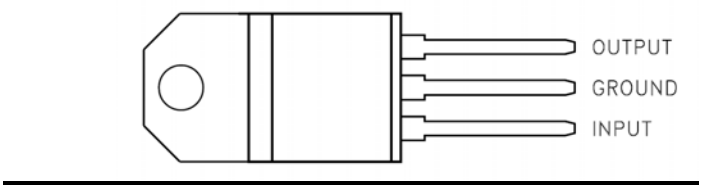
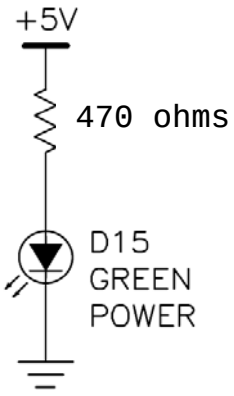


Figure 2 - PICkit 1 Starter Kit Power Indication Circuit



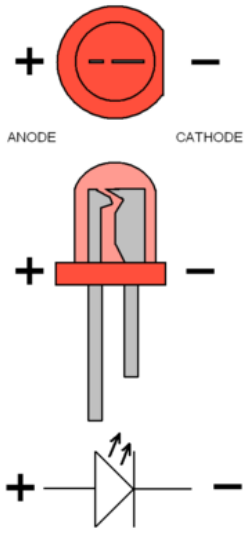


Figure 3 - PICkit 1 Starter Kit LED Layout

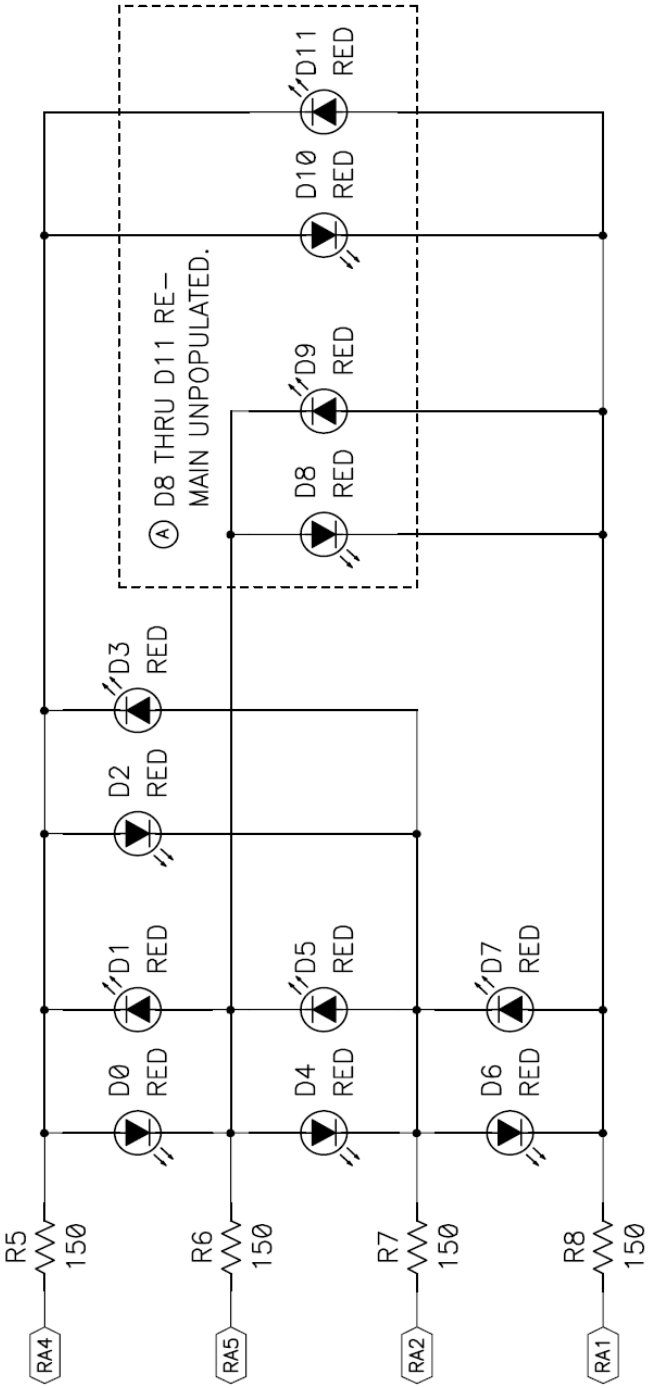


Figure 4 – PICkit 1 Starter Kit ADC Input Circuit

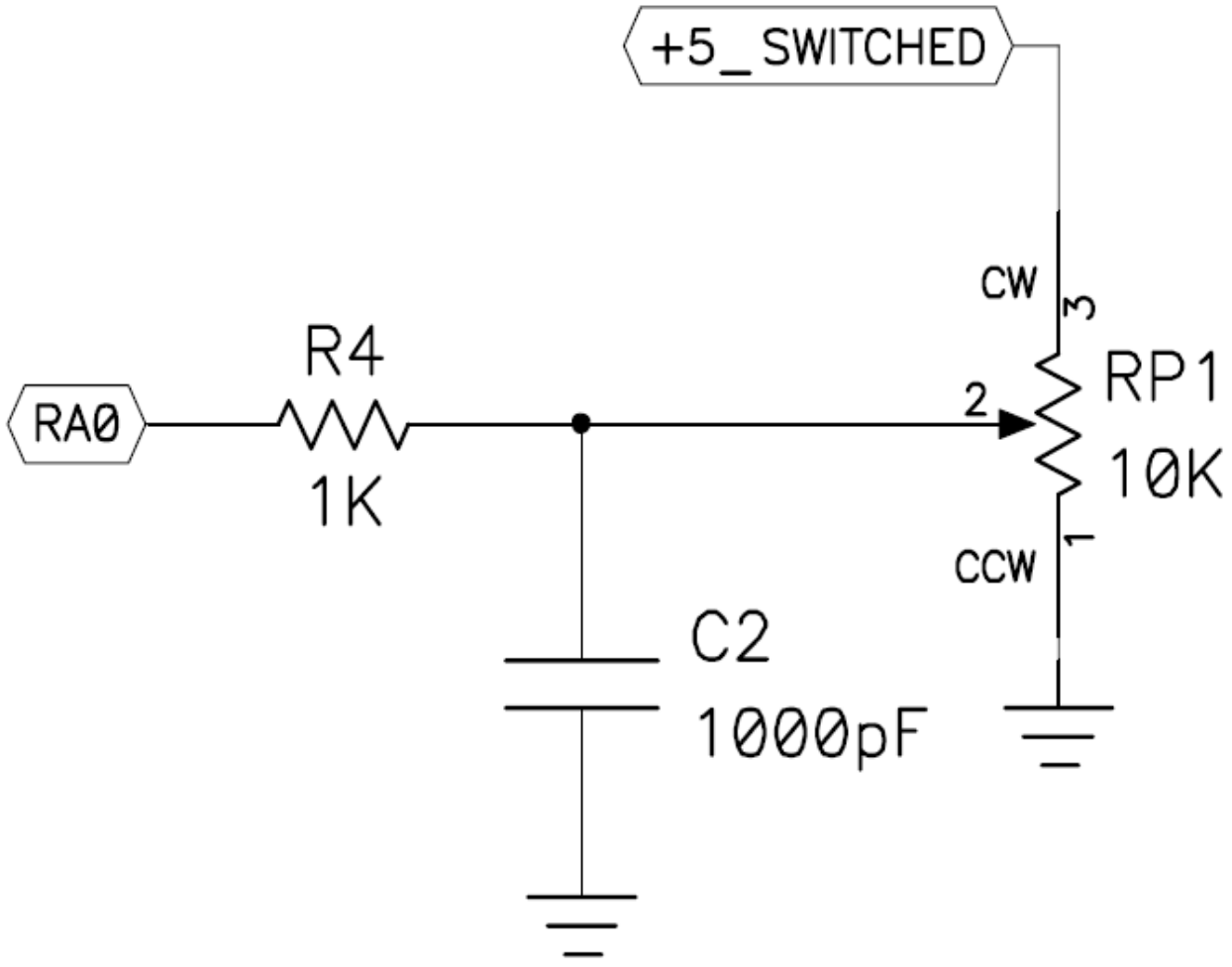


Figure 5 Lab_4A.c Code

```
#include <pic.h>


/*****************************************************
* Function: main
*
* Description:  D0 - D7 on PICkit 1 will Display the results of the ADC
*
* Notes:
*
* RA0 - Input from RP1
*
*
*
* Returns:  This routine contains an infinite loop
*
*****************************************************/

/* Configuration Word */

__CONFIG(INTIO & WDTDIS & PWRTEN & MCLRDIS & UNPROTECT \
  & UNPROTECT & BORDIS & IESODIS & FCMDIS);


void PORTA_init(void);
void ADC_Disp(void);
void Delay_LED_On(void);

int ADC_Value = 0;

const char PORTA_Value[8] = {
      0b010000,   // D0
      0b100000,   // D1
      0b010000,   // D2
      0b000100,   // D3
      0b100000,   // D4
      0b000100,   // D5
      0b000100,   // D6
      0b000010};  // D7

const char TRISA_Value[8] = {
      0b001111,   // D0
      0b001111,   // D1
      0b101011,   // D2
      0b101011,   // D3
      0b011011,   // D4
      0b011011,   // D5
      0b111001,   // D6
      b111001};   // D7
```

Continued next page

Continued from previous page

```
main()
{
    PORTA_init();

    ANSEL = 1;                  //  Just RA0 is an Analog Input
    TRISA0 = 1;                 //  Corresponding TRIS bit is set as input

    ADCON0 = 0b00000001;        //  Turn on the ADC
                                //   Bit 7          - Left Justified Sample
                                //   Bit 6          - Use VDD
                                //   Bit 4:2        - Channel 0
                                //   Bit 1          - Do not Start
                                //   Bit 0          - Turn on ADC

    ADCON1 = 0b00010000;        //  Select the Clock as Fosc/8


    ADC_Disp();
    GODONE = 1;                 // Start A/D Conversion


    while(1 == 1)               //  Loop Forever
    {

        if (GODONE == 0)                // Is A/D Conversion complete?
            {     ADC_Disp();           // Display A/D Conversion Results
                  ADC_Value = ADRESH;   // Get new A/D value
                  GODONE = 1;           // Start the next A/D Conversion
              }
          else                          // A/D Conversion still in progress
                  ADC_Disp();

      }
}

/******** END OF main ROUTINE *************************/

/***************************************************
* Function: PORT_init
*
* Description:  Initializes PORTA to a known condition
*
* Notes:    None
*
* Returns:  None
*
***************************************************/
void PORTA_init(void)
{
      PORTA = 0;                //  All PORTA Pins are low
      CMCON0 = 7;               //  Turn off Comparators
      ANSEL = 0;                //  Turn off ADC

      return;

}
/******** END OF PORTA_init ***************************/
```

Continued next page

Continued from previous page

```
/*******************************************************
* Function: ADC_Disp
*
* Description:  Displays the value of A/D Conversion on D0 - D7
*
* Notes:
*
*
*
* Returns:  None
*
*******************************************************/
void ADC_Disp(void)
{

      int i;

      for (i = 0; i < 8; i++ )
      {                              // Loop through Each of the 8 LEDS

        Delay_LED_On();              // Allows time for individual LEDs to light

            if ((ADC_Value & (1 << i)) == 0)
                PORTA = 0;
            else
                PORTA = PORTA_Value[i];
                TRISA = TRISA_Value[i];
        }  //

      return;

}
/******** END OF ADC_Disp ************************/

/*******************************************************
* Function: delay_LED_On
*
* Description:  Causes a delay in program execution
*
* Notes:
*
* Delay was determined through trial and error
*
* Returns:  None
*
*******************************************************/
void Delay_LED_On(void)
{
      int j;

      for (j = 0; j < 60; j++);           //  Display "On" Loop

      return;

}
/******** END OF Delay_LED_On ************************/
```

Figure 6 Required Header Code

```
/*******************************************************
*
* Program Title:     xxxx
*
* Program File Name: Lab_XA.c
*
* Microprocessors A 17.383
*
* xxxxxxxx - Put in Semester (i.e. Fall 2010) here
*
* xxxxxxxx - Put in your name here
*
* xx/xx/xx - Put date here
*
*******************************************************/
```